Universidad de Buenos Aires - FIUBA
66.20 Organización de Computadoras
Trabajo práctico 1: conjunto de instrucciones MIPS
1^{er} cuatrimestre de 2018

`$Date: 2018/04/10 22:05:55 $`

1. Objetivos

Familiarizarse con el conjunto de instrucciones MIPS y el concepto de ABI, extendiendo un programa que resuelva el problema descripto en la sección 4.

2. Alcance

Este trabajo práctico es de elaboración grupal, evaluación individual, y de carácter obligatorio para todos alumnos del curso.

3. Requisitos

El trabajo deberá ser entregado personalmente, en la fecha estipulada, con una carátula que contenga los datos completos de todos los integrantes, un informe impreso de acuerdo con lo que mencionaremos en la sección 5, y con una copia digital de los archivos fuente necesarios para compilar el trabajo.

4. Descripción

Se trata de un modificar un programa que dibuje el conjunto Julia de y sus vecindades, en el cual la lógica de de cómputo del fractal deberá tener soporte nativo para NetBSD/pmax.

El código fuente con la versión inicial del programa, se encuentra disponible en [7]. El mismo deberá ser considerado como punto de partida de todas las implementaciones.

4.1. Programa

El programa recibe, por línea de comando, la descripción de la región del plano complejo y las características del archivo imagen a generar. No interactúa con el usuario, ya que no se trata de un programa interactivo, sino más bien de una herramienta de procesamiento *batch*. Al finalizar la ejecución, y volver al sistema operativo, el programa habrá dibujado el fractal en el archivo de salida.

El formato gráfico a usar es PGM o *portable gray map* [6], un formato simple para describir imágenes digitales monocromáticas.

4.2. Algoritmo

El algoritmo básico es simple: para algunos puntos f_0 de la región del plano que estamos procesando haremos un cálculo repetitivo. Terminado el cálculo, asignamos el nivel de intensidad del pixel en base a la condición de corte de ese cálculo.

El color de cada punto representa la "velocidad de escape" asociada con ese número complejo: blanco para aquellos puntos que pertenecen al conjunto (y por ende la "cuenta" permanece acotada), y tonos gradualmente más oscuros para los puntos divergentes, que no pertenezcan al conjunto.

Más específicamente: para cada pixel de la pantalla, tomaremos su punto medio, expresado en coordenadas complejas, $f_0 = \mathrm{Re}(f_0) + \mathrm{Im}(f_0)i$. A continuación, iteramos sobre $f_{i+1}(c) = f_i(c)^2 + s$, cortando la iteración cuando $|f_i(c)| > 2$, o después de `N` iteraciones.

En pseudo código:

```
para cada pixel $p {
        $f = complejo asociado a $p;
        for ($i = 0; $i < $N - 1; ++$i) {
                if (abs($f) > 2)
                        break;
                $f = $f * $f + $s;
        }
        dibujar el punto p con brillo $i;
}
```

Aquí, el parámetro s representa la semilla o *seed* usada para generar el fractal. Se trata de una constante compleja que nos permite parametrizar la forma del fractal, cuyo valor por defecto está especificado en la sección 4.3.

Así tendremos, al finalizar, una representación visual de la cantidad de ciclos de cómputo realizados hasta alcanzar la condición de escape (ver figura 1).

4.3. Interfaz

A fin de facilitar el intercambio de código *ad-hoc*, normalizaremos algunas de las opciones que deberán ser provistas por el programa:

- `-m`, o `--method`, permite seleccionar dinámicamente la implementación a usar para generar la imagen de salida: `generic` para seleccionar la implementación genérica, en lenguaje C; `mips32` para activar el soporte nativo para procesadores MIPS. Por defecto, el programa usa `generic`.
- `-r`, o `--resolution`, permite cambiar la resolución de la imagen generada. El valor por defecto será de `640x480` puntos.
- `-c`, o `--center`, para especificar las coordenadas correspondientes al punto central de la porción del plano complejo dibujada, expresado en forma binómica (i.e. $a+bi$). Por defecto usaremos `0+0i`.
- `-w`, o `--width`, especifica el ancho de la región del plano complejo que estamos por dibujar. Valor por defecto: `2`.

- `-H`, o `--height`, sirve, en forma similar, para especificar el alto del rectángulo a dibujar. Valor por defecto: `2`.
- `-s`, o `--seed`, para configurar el valor complejo de la semilla usada para generar el fractal. Valor por defecto: `-0.7268953477091140+0.1888871290438459i`.
- `-o`, o `--output`, permite colocar la imagen de salida, (en formato PGM [6]) en el archivo pasado como argumento; o por salida estándar -`cout`- si el argumento es "-".

4.4. Soporte para MIPS

El entregable producido en este trabajo deberá implementar la lógica de cómputo del fractal en assembly MIPS, con soporte nativo para NetBSD/pmax.

Para ello, cada grupo deberá tomar el código fuente de base para este TP, [7], y reescribir la función `mips32_plot()` sin cambiar su API. Esta función está ubicada en el archivo `mips32_plot.c`.

4.5. Casos de prueba

Es necesario que el informe trabajo práctico incluya una sección dedicada a verificar el funcionamiento del código implementado.

En el caso del TP 0, será necesario escribir pruebas orientadas a probar el programa completo, ejercitando los casos más comunes de funcionamiento, los casos de borde, y también casos de error.

Incluimos en este enunciado dos fractales de referencia que pueden ser usados para comprobar visualmente el funcionamiento del programa.

4.6. Ejemplos

Generamos un dibujo usando los valores por defecto, barriendo la región rectangular del plano comprendida entre los vértices $-1+1i$ y $+1-1i$.

```
$ tp0 -m mips32 -o uno.pgm
```

La figura 1 muestra la imagen `uno.pgm`.

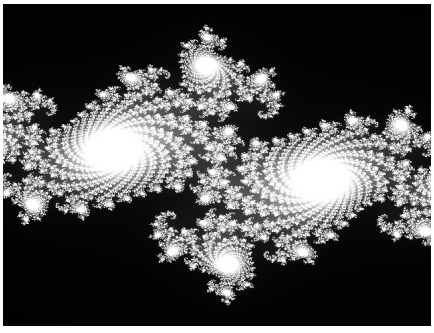

Figura 1: Región barrida por defecto.

A continuación, hacemos *zoom* sobre la región centrada en `0.282-0.007i`, usando un rectángulo de `0.005` unidades de lado.

```
$ tp0 -m mips32 -c 0.282-0.007i -w 0.005 -H 0.005 -o dos.pgm
```

El resultado podemos observarlo en la figura 2.

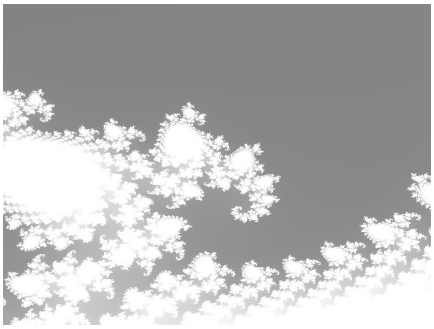

Figura 2: Región comprendida entre $0{,}2795 - 0{,}0045i$ y $0{,}2845 - 0{,}0095i$.

5. Informe

El informe deberá incluir:

- Documentación relevante al diseño e implementación del programa.
- Documentación relevante al proceso de compilación: cómo obtener el ejecutable a partir de los archivos fuente.
- Las corridas de prueba, con los comentarios pertinentes.
- El código fuente completo, en lenguaje C y MIPS32. Esto incluye el código MIPS32 escrito en forma manual, no es necesario incluir la salida del compilador.
- Este enunciado.

6. Fecha de entrega

La última fecha de entrega y presentación sería el martes 8/5.

Referencias

[1] GXemul, `http://gavare.se/gxemul/`.

[2] The NetBSD project.
`http://www.netbsd.org/`.

[3] `http://es.wikipedia.org/wiki/Conjunto_de_Julia` (Wikipedia).

[4] `http://mathworld.wolfram.com/JuliaSet.html` (Mathworld).

[5] Smooth shading for the Mandelbrot exterior.
`http://linas.org/art-gallery/escape/smooth.html`. Linas Vepstas. October, 1997.

[6] PGM format specification.
`http://netpbm.sourceforge.net/doc/pgm.html`.

[7] Código fuente con el esqueleto del trabajo práctico.
`http://www.fiuba7504.com.ar/tp1-2014-2-src.tar.gz`.